Topological Visualization Method for Understanding the Landscape of Value Functions and Structure of the State Space in Reinforcement Learning

Yuki Nakamura[1][a] and Takeshi Shibuya[2][b]

[1]*Graduate School of Systems and Information Engineering, University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Japan*

[2]*Faculty of Engineering, Information and Systems, University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Japan*

Keywords: Reinforcement Learning, Topological Data Analysis, TDA Mapper, Visualization.

Abstract: Reinforcement learning is a learning framework applied in various fields in which agents autonomously acquire control rules. Using this method, the designer constructs a state space and reward function and sets various parameters to obtain ideal performance. The actual performance of the agent depends on the design. Accordingly, a poor design causes poor performance. In that case, the designer needs to examine the cause of the poor performance; to do so, it is important for the designer to understand the current agent control rules. In the case where the state space is less than or equal to two dimensions, visualizing the landscape of the value function and the structure of the state space is the most powerful method to understand these rules. However, in other cases, there is no method for such a visualization. In this paper, we propose a method to visualize the landscape of the value function and the structure of the state space even when the state space has a high number of dimensions. Concretely, we employ topological data analysis for the visualization. We confirm the effectiveness of the proposed method via several numerical experiments.

1 INTRODUCTION

Reinforcement learning, in the field of machine learning is a learning framework that autonomously acquires control rules to maximize the rewards from the environment via the trial and error of agents (Sutton and Barto, 1998; Matarić, 1997). As a result, it is possible to lighten the burden on designers of designing complex algorithms (Smart and Pack Kaelbling, 2002).

When acquiring control rules via reinforcement learning, the designer needs to set up the environment, learning methods, and parameters; the environment consists of the state spaces and reward functions, and the parameters include the learning rates, rates of exploration and exploitation, and discount rates. Agents learn by repeated exploration and exploitation under given circumstances. Therefore, the performance of the acquired control rules changes depending the design. If the performance required by the designer cannot be obtained, it is necessary to investigate the cause of this failure. Then, the environment, learning method and parameters are reconstructed and reinforcement learning is performed. By repeating this cycle, the agent aims to gain the performance required by the designer. In other words, it is necessary to have a method to investigate causes when the required performance cannot be obtained.

For such an investigation, it is necessary to understand how the learning is being performed at that time, that is, it is important to understand the control rules of the agent. There are several ways to determine this. Commonly used methods include examining the sum of rewards acquired for each episode, examining the control rules of the agent heuristically, and examining the landscape of the value function and the structure of the state space. Even though the learning progress can be understood via a method examining the total sum of rewards acquired for each episode, it is impossible to know why the required performance cannot be obtained if the rewards cannot be acquired. In the heuristic method, when the state space is complicated, there are many situations that need to be examined, which requires a great deal of trials and errors. In the case where the state space is less than or equal to two dimensions, visualizing the landscape of the

[a] https://orcid.org/0000-0001-9922-890X
[b] https://orcid.org/0000-0003-4645-5898

Nakamura, Y. and Shibuya, T.
Topological Visualization Method for Understanding the Landscape of Value Functions and Structure of the State Space in Reinforcement Learning.
DOI: 10.5220/0008913303700377
In *Proceedings of the 12th International Conference on Agents and Artificial Intelligence (ICAART 2020) - Volume 2*, pages 370-377
ISBN: 978-989-758-395-7; ISSN: 2184-433X

value function and the structure of the state space is the most powerful method to understand the control rules of the agent. This is because, we can visually understand the progress of the explore and the evaluation of the state. In this paper, we aim to be able to understand the control rules of the agent by focusing on the landscape of the value function and the structure of the state space. When the state space is low dimensional, visualization techniques can represent the landscape of the value function and the structure of the state space; however, when the state space is high dimensional, it becomes difficult to visualize.

Therefore, the purpose of this study is to visualize the landscape of the value function and the structure of the state space when the state space is high dimensional. Additionally, we show that this visualization is useful for understanding the control rules of the agent in reinforcement learning.

2 BACKGROUND

Reinforcement learning involves learning motion selection for a certain state to maximize the reward (Sutton and Barto, 1998). Many algorithms for reinforcement learning are based on estimating value functions. These function are classified into two types: state value functions and action value functions. A state value function is represented by $V(s)$ and evaluates how much value an agent has in a given state. An action value function is represented by $Q(s,a)$ and evaluates how much value is gained by performing a given action in a given state for an agent.

2.1 Q-learning

Q-learning (Watkins and Dayan, 1992) is a value-updating algorithm in reinforcement learning that updates the action value function $Q(s_t,a_t)$ using Equation (1).

$$Q(s_t,a_t) \leftarrow (1-\alpha)Q(s_t,a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1},a) \right] \tag{1}$$

Here, α is a parameter called the step-size, where $0 \leq \alpha \leq 1$. This parameter controls the rate at which the action value function $Q(s_t,a_t)$ is updated.

The ε-greedy method is often used as a behavioral selection method in Q-learning, where $0 \leq \varepsilon \leq 1$. This is a method of taking random action with a probability of ε and taking the action probability function with the largest action with the probability of $1-\varepsilon$. In other words, the larger the value of ε, the higher the exploration rate, and the smaller the value of ε, the higher the exploitation rate.

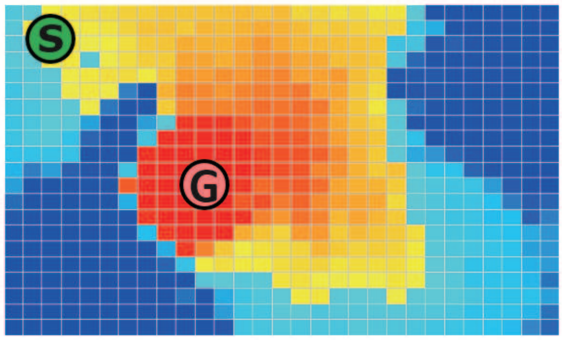

Figure 1: A heat map of learning how to climb a virtual volcano from S (the start) to G (the goal) at the summit.

2.2 Heat Map

A heat map can be used as a method to visualize the landscape of the value function and the structure of the state space; an example of heat map is shown in Figure 1. Here, a two-dimensional state space is defined in the domain and the state value function is represented by the color shading. The heat map in Figure 1 represents a learning of how to climb a virtual volcano from S (the start) to G (the goal) at the summit. In general, an agent has a policy to perform a state transition to a state higher than the state value of the existing state. In other words, when visualized with this heat map, the transition to a warmer color state is taken to be the policy and the probability of repeating the state transition to a warmer color state where a transition is possible is high. By analyzing the heat map, the designer can roughly understand how the agent repeats the state transition. Therefore, the heat map visualizes the structure of the state space and the landscape of the value function and is useful for understanding the agent control rule.

However, it is rare for a heat map to be able to visualize the structure the of state space and the landscape of the value function. This is because a heat map has only two dimensions in the state space and it is difficult to use it in the case of three or more dimensions. Moreover, in the example of the virtual volcano in Figure 1, it is assumed that the state transition can be performed in the adjacent state; however, in actuality, the state transition cannot be performed in the adjacent state nor can the state transition be performed in the non-adjacent state. If a system that can visualize the structure of the state space and the landscape of the value function exists, even if such a state space has high dimensions or the structure of the state space is complex, it would be useful. This is the purpose of this study.

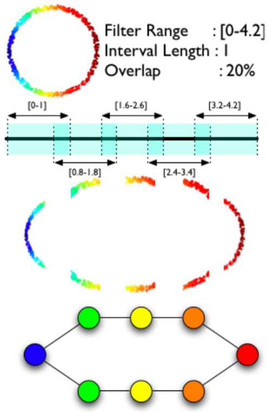

Figure 2: The TDA Mapper (from Singh et al. (Singh et al., 2007) Figure 1).

2.3 TDA Mapper

In recent years, the TDA Mapper has attracted attention as a method for analyzing high dimensional and large-scale data and as a big data analysis technology that makes it easier for users to understand data (Singh et al., 2007; Hamada and Chiba, 2017; Wang et al., 2015). This is a method to topologically analyze data and create a graph with a topological structure similar to the original data. TDA Mapper mainly analyzes point clouds, and uses the distance matrix of point clouds as input data. The topological analysis of high dimensional data is the focus of this study.

Figure 2 shows how to create a graph using TDA Mapper. This is an example of creating a graph with circularly distributed data while maintaining the topological structure. The graph creation algorithm in particular is shown. We analyze a point cloud

$$X = \{x_1, \ldots, x_n\}.$$

The point cloud X is distributed in a circle.

1. By defining the filter function,

   $$h : X \to \mathbb{R},$$

   on the point cloud X, a filter value $h(x)$ is given to each point x. The circle is given a filter value that increases from left to right.

2. The range $h(X)$ of the filter function is covered with m intervals, $I_1, \ldots, I_m$, and the point group X is divided into level sets:

   $$X_i = \{x \in X \mid h(x) \in I_i\}.$$

   Here, the lengths of the divided intervals are all equal and the adjacent intervals I_i and I_{i+1} both overlap by $p\%$ of the interval length. The example in the Figure 2 is divided into five intervals with 20% overlap.

3. Each level set X_i is clustered using an arbitrary method and divided into clusters:

   $$X_i = \bigoplus_j X_i^j,$$

   where $\bigoplus$ stands for direct sum and is synonymous with

   $$X_i = \bigcup_j X_i^j,\ X_i^j \bigcap X_i^{j'} = \phi\,(j \neq j').$$

   The adjacency relationships of the clusters X_i^j are expressed as a graph. Specifically, with a cluster as a vertex, a graph

   $$G_h(X) = \begin{cases} V = \{X_i^j\} \\ E = \{(X_i^j, X_{i+1}^k) \mid X_i^j \bigcap X_{i+1}^k \neq \phi\} \end{cases}$$

   is constructed in which the edges are extended between the overlapping clusters belonging to adjacent levels. In Figure 2, the vertices of the graph are ordered according to the level.

TDA Mapper requires the user to set many parameters at the time of graph creation; These parameters are the number of divisions of the filter function range, the overlap ratio, and the clustering method. From this, depending on the settings of the parameters, the shape of the graph can differ greatly and it is difficult to determine whether the output graph is useful. The proposed method aims to improve this problem as well.

3 THE PROPOSED METHOD: RL MAPPER

We propose a method to visualize the structure of the state space and the landscape of the value function even when the state space is high dimensional and complex. The proposed method is named RL Mapper, where RL is an abbreviation for reinforcement leaning. In this method, we modified the 2.3 TDA Mapper algorithm to be appropriate for reinforcement learning. TDA Mapper analyzes point clouds and uses data on the distance between points. On the other hand, RL Mapper analyzes the landscape of the value function and the structure of the state space. We make it possible to represent the landscape of the value function by using state value data. Additionally, we make it possible to define the distance between states by using the history of agent state transitions.

Here, we describe the specific algorithm. We visualize the landscape of the value function and the structure of the state space of the state set,

$$S = \{s_1, \ldots, s_n\},$$

which is completely learned or under learning.

1. We define the filter function as the state value function

$$v : S \to \mathbb{R}.$$

We give a filter value $v(s)$ for each state $s \in S$.

2. We define m uncovering intervals $I_1, \ldots, I_m$ in the range $v(s)$ of the filter function. We divide the state set S into level sets:

$$S_i = \{s \in S \mid v(s) \in I_i\}.$$

Here, the sizes of the sections $I_1, \ldots, I_m$ are equal.

3. We cluster each level set S_i and divide it into clusters:

$$S_i = \bigoplus_j S_i^j.$$

Here, we describe the clustering method used in this approach. We define a set of states having a history of state transitions between the states $s \in S_i$ of the level set S_i as one cluster. We construct a graph

$$G_v(S) = \begin{cases} V = \{S_i^j\} \\ E = \{(S_i^j, S_k^l) \mid S_i^j \text{ and } S_k^l \text{ have one} \\ \quad \text{or more pairs of state pairs } S_i^j \text{ and } S_k^l \\ \quad \text{that can transition between states}\} \end{cases}$$

with nodes as clusters and edges defined below.

We call this graph, $G_v(S)$, the topological state graph. We show an example of a topological state graph in Figure 3.

The features of the topological state graph are as follows.

- A node is a set of “state value close” and “state transitionable” states.
- The color density of a node is the average of the state value of the state clustered on that node.
  - A warm color indicates a high state value.
  - A cool color indicates a low state value.
- We give the node a number x-y.
  - x is a number indicating the interval of the state value, where the state value is larger if the number is larger.
  - y is the cluster label within the x interval.
- The size of a node is correlated to the number of states included in that node.
- Between nodes connected by an edge, there are one or more state pairs capable of a state transition.

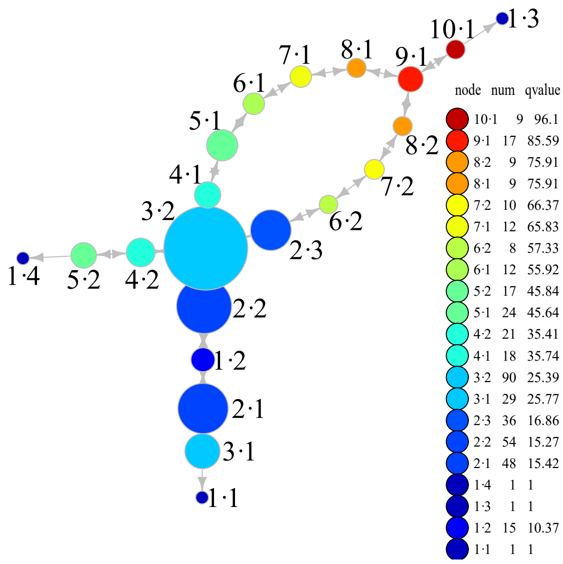

Figure 3: Topological stage graph.

The inputs to RL Mapper are state values of each state and adjacency matrix representing the history of state transitions based on agent experience. The only parameter set by the user is the number of divisions of the filter value for the state value function. In reinforcement learning, it is difficult to define the distance between states; therefore, we performed clustering using the history of the state transitions. This makes it possible to store the structure of the state space with higher accuracy. Moreover, because the number of parameters is small, it is easy for the designer to understand how the graph visually changes owing to adjustments to the parameters.

4 EXPERIMENTS

The topological state graph, which visualizes the landscape of the value function and the structure of the state space with RL Mapper, is shown via experiments to be useful for understanding the agent control rules. We experimented in two situations. In Experiment 1, the state space is a two-dimensional path search problem, and in Experiment 2 the state space is a four-dimensional taxi task. In Experiment 1, by comparing the heat map with the topological state graph, we show that the topological state graph reproduces the utility of the heat map. We divided Experiment 1 into 1-A and 1-B in which only the value of reward was changed. In this way, we show that by visualizing the landscape of the value function and the structure of the state space, it is possible for the designer to understand the difference in the control rules of the agent caused by the change of the reward function. In Experiment 2, we show that RL Mapper is useful in the case of the high dimensional state space as well as in the low dimensional one.

4.1 Two-dimensional Path Search Problem

4.1.1 Outline of the Experiment

We divided the two-dimensional path search problem into Experiment 1-A and 1-B. Figure 4 shows the environment of the route search problem used in Experiment 1-A, and Figure 5 shows the same for Experiment 1-B. In both experiments, 1-A and 1-B, the agent is on a two-dimensional plane that satisfies $x, y \in \mathbb{Z},\ 0 \leq x, y \leq 21$. The state space is two-dimensional consisting of the x coordinates and y coordinates. The agent can recognize only the current coordinates and the obtained reward and cannot recognize other environmental information.

The agent, as an action can move one square in the vertical or horizontal direction. The agent cannot move while in the state partitioned by the wall (the thick line) and remains in the original state when selecting an action to move toward the wall. Experiment 1-A starts at S, and when the agent reaches a point indicated as 30, 50, 1000, or -500, we reward the agent with 30, 50, 1000, or -500, respectively. Additionally, when the agent reaches a point indicated as 30, 50, or 1000, the episode ends and the next episode starts from the point of S. Similarly, if Experiment 1-B starts at S, and the agent reaches the points indicated as 30, 50, 100, or -40, we reward the agent with 30, 50, 100, or -40, respectively. When the agent reaches a point indicated as 30, 50, or 100, the episode ends and the next episode starts from the point S.

We use Q-learning and the set step-size $\alpha = 0.5$, explore ratio $\varepsilon = 0.999^n$, where n is the number of episodes and the discount rate $\gamma = 0.95$. The number of divisions in the range of the filter function, which is a parameter of the RL Mapper, is 10.

4.1.2 Results and Discussion

We show the heat map and state phase graph of 10,000 episodes in Experiment 1-A in Figures 6 and 7, respectively. The labels of the nodes correspond to the labels of the heat map.

The graphs of the 10,000 episodes shown in Figures 6 and 7 are for after the learning has been completed. The agent has a policy of repeating state transitions to higher states of the state value function. Therefore, we can see from the heat map in Figure

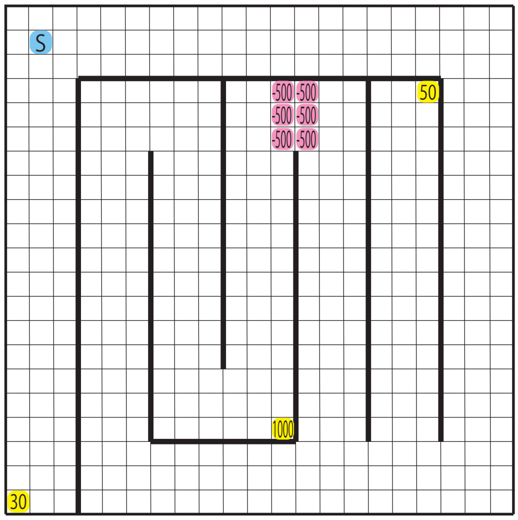

Figure 4: Environment of Experiment 1-A.

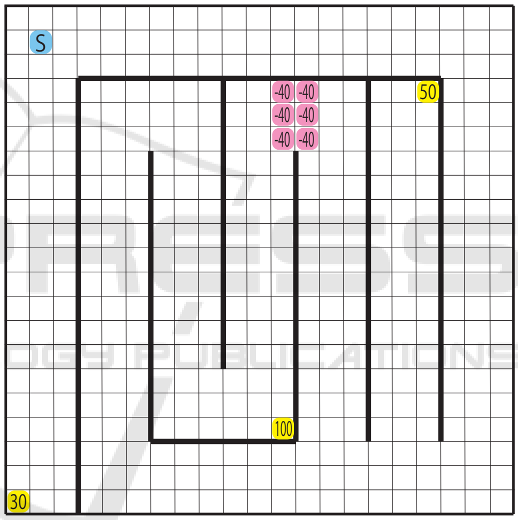

Figure 5: Environment of Experiment 1-B.

6 that the agent has a state transition from 2-1 to 3-3 to 4-1 to 5-1 to 6-2 to 7-1 to 8-1 to 9-1 to 10-1 to 1-2 as the control rule. As an exception, in 1-2, the state value remains low because the agent receives a reward and the episode ends. That is, the agent aims to reach the state where it can earn a reward of 1000. Here, there is one other way to reach the state where the agent can earn a reward of 1000. It is a path where the agent transitions from 2-1 to 3-3 to 4-1 to 3-1 to 7-2 or 8-2 to 9-2 to 10-1 to 1-2. However, the agent does not have this route as the control rule because the state value decreases at the state transition from 4-1 to 3-1. Similarly, we can see from the topological state graph in Figure 7 that the agent has a state transition from 2-1 to 3-3 to 4-1 to 5-1 to 6-2 to 7-1 to 8-1 to 9-1 to 10-1 to 1-2 as the control rule. The topological state graph also shows another way to reach the state

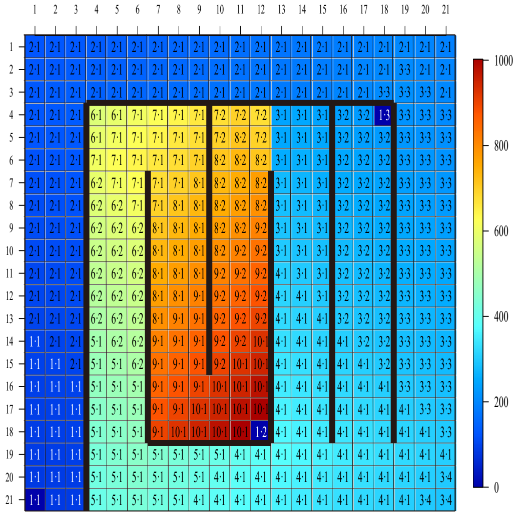

Figure 6: Heat map for 10,000 episodes in Experiment 1-A.

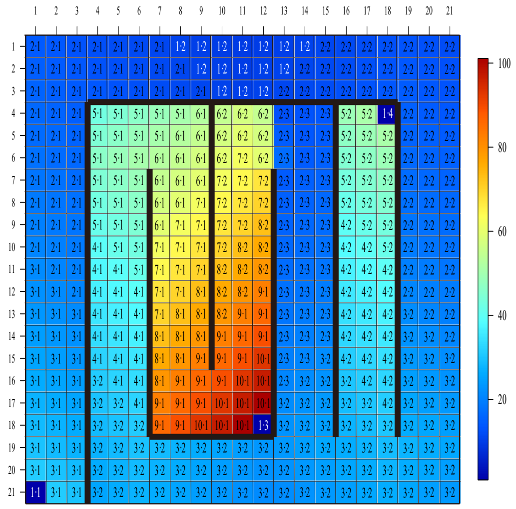

Figure 8: Heat map for 10,000 episodes in Experiment 1-B.

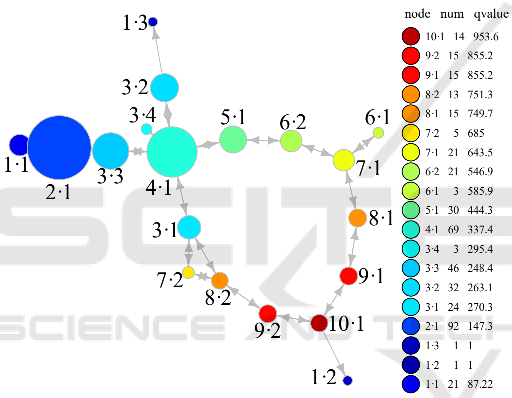

Figure 7: Topological state graph for 10,000 episodes in Experiment 1-A.

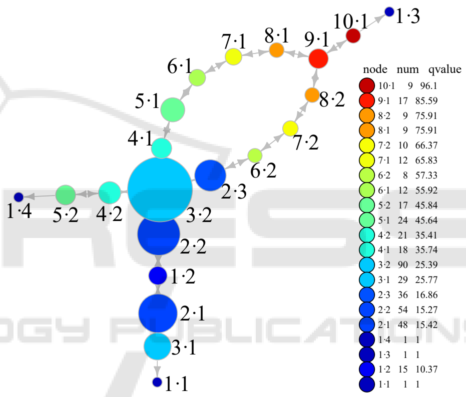

Figure 9: Topological state graph for 10,000 episodes in Experiment 1-B.

where the agent can earn a reward of 1000. we can see that the agent does not have this route as the control rule because the state value decreases at the state transition from 4-1 to 3-1.

Next, we show the heat map and state phase graph of the 10,000 episodes in Experiment 1-B in Figures 8 and 9, respectively. The graphs of the 10,000 episodes shown in Figure 8 and 9 are for after the learning has been completed. We can see from the heat map in Figure 8 that the agent has a state transition from 2-1 to 3-1 to 1-1 as the control rule. That is, the agent aims to reach the state where it can earn a reward of 30. The agent does not aim to reach a state where it can earn rewards of 50 or 100. This is because this path has a reduced state value at the state transition from 2-1 to 1-2. Similarly, we can see from the topological state graph in Figure 9 that the agent has a state transition from 2-1 to 3-1 to 1-1 as the control rule.

From Experiments 1-A and 1-B, we can see that the topological state graph output by RL Mapper preserves the landscape of the value function and the structure of the state space. Additionally, we find that the agent control rules are different when comparing Experiments 1-A and 1-B. The difference in the control rules results from the difference in the design of the reward function. The designer can visually understand the differences in control rules using the topological state graph. Therefore, topological state graphs are useful for understanding the control rules of agents.

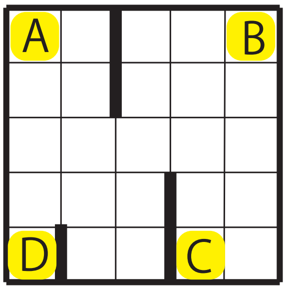

Figure 10: The taxi task.

4.2 Four-dimensional Taxi Task

4.2.1 Outline of the Experiment

The taxi task is to pick up a passenger in the environment shown in Figure 10 and carry them to the destination (Şimşek et al., 2005). The agent here is a taxi. A taxi, a passenger are located on a two-dimensional plane that satisfies $x, y \in \mathbb{Z},\ 0 \leq x, y \leq 5$. The locations and destinations of the passengers are randomly selected from the locations indicated by A, B, C, and D. The initial position of the taxi is randomly selected from the positions satisfying the above equation. At each location, taxis have one choice of action: move one square north, south, east, or west, pick up a passenger, or drop off a passenger. The action of picking up a passenger is only possible if the passenger is in the same position as the taxi. Similarly, dropping off a passenger is only possible if the passenger is in the taxi and the taxi is at the destination. additionally, agents cannot transition between pairs of states separated by walls (the thick line). If the agent chooses to move toward a wall, it remains in its original state. We reward the agent with 20 if the agent can deliver the passenger to the destination. Additionally, when we reward an agent, we end that episode and move to the next episode. The state space is four-dimensional consisting of the x coordinates, y coordinates, passenger status, and destination location.

We use Q-learning and the set step-size $\alpha = 0.5$, the explore ratio $\varepsilon = 0.3$, and the discount rate $\gamma = 0.9$. The number of divisions in the range of the filter function, which is a parameter of the RL Mapper, is 6.

4.2.2 Results and Discussion

We show the state phase graph for 5000 and 10,000 episodes in the Figure 11 and 12. RL Mapper can also be used on the data during learning. The 5000 episodes are for the data during the learning, and the 10,000 episodes are for the data after the learning. We compared the topological state graphs for the 5000 episodes in Figure 11 to the 10,000 episodes in Figure 12. We can see that the topological state graph for the 10,000 episodes has fewer nodes and edges and that the graph is simpler than that for 5000 episodes. This is because the learning process has smoothened the landscape of the values. Additionally, we can see that the graph is composed of four groups in the topological state graphs for 5000 and 10,000 episodes. This is because there are four destinations, which are separate tasks, and there is no state transition between tasks.

Considering the topological state graph for 10,000 episodes in the Figure 12, we find that there are three routes going to the high-value nodes 6-1, 6-2, 6-3, and 6-4, and that these routes join to reach high-value nodes. This is due to the three locations of the passengers not boarding at each destination. When a passenger takes a taxi, they transition to high-value nodes and reach nodes 1-1, 1-3, 1-6 and 1-8, where rewards can be obtained.

Therefore, in the case of high dimensional state space, RL Mapper can visualize the landscape of the value function and the structure of the state space. Moreover, we were able to understand the rough control rule of the agent using the topological state graph.

5 CONCLUSIONS

The purpose of this paper is to allow the designer to understand the agent control rules by visualizing the landscape of the value function and the structure of the state space when the state space is high dimen-

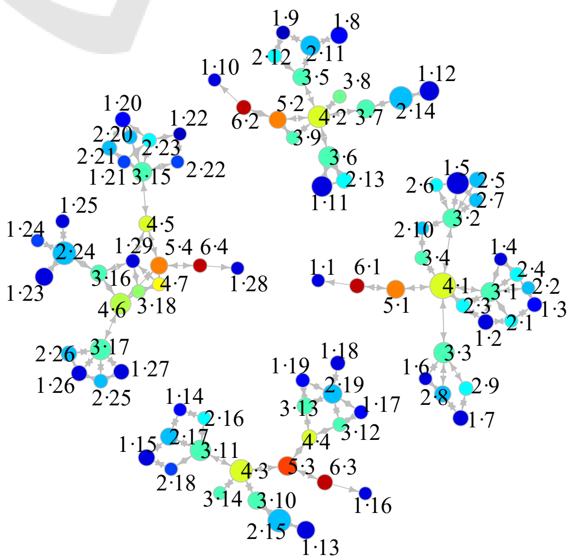

Figure 11: Topological state graph for 5000 episodes.

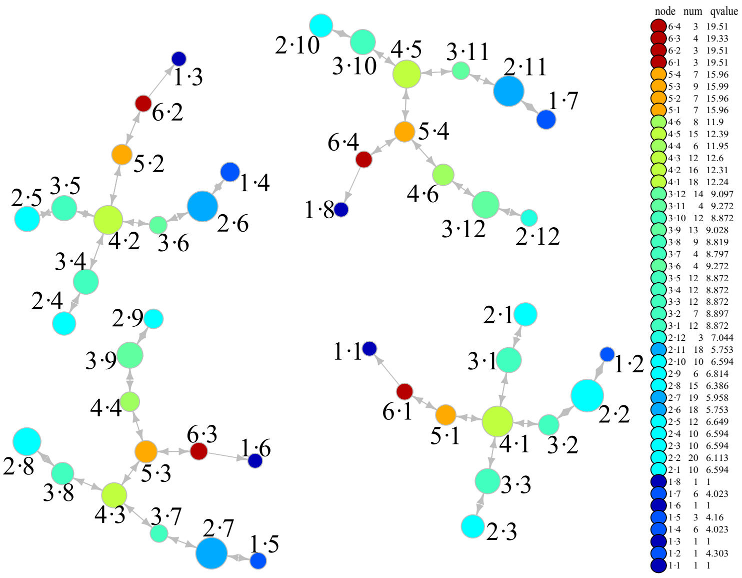

Figure 12: Topological state graph for 10,000 episodes.

sional. When the state space is low dimensional, a method of visualizing the landscape of the value function and the structure of the state space already exists, the heat map; however, when the state space is high dimensional, there is no method for such a visualization. Therefore, we proposed RL Mapper, a visualization method that focuses on the topological structure of the data. We examined the correspondence between a heat map and a topological state graph when the state space was two dimensional using a path search problem and showed that the topological state graph retains the usefulness of a heat map. We also showed that the visualization of the value function landscape and the state space structure in RL Mapper is useful for understanding the agent control rules. Additionally, using the taxi task, we showed that RL Mapper can provide the same visualization even when the state space is four dimensional. Therefore, RL Mapper can visualize the landscape of the value function and the structure of the state space in the case of a high dimensional state space. We also demonstrated that this visualization is useful for understanding the control rules of agents.

REFERENCES

Şimşek, O., Wolfe, A. P., and Barto, A. G. (2005). Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 816–823, New York, NY, USA. ACM.

Hamada, N. and Chiba, K. (2017). Knee point analysis of many-objective pareto fronts based on reeb graph. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 1603–1612.

Matarić, M. J. (1997). Reinforcement learning in the multi-robot domain. *Auton. Robots*, 4(1):73–83.

Singh, G., Memoli, F., and Carlsson, G. (2007). Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition. In Botsch, M., Pajarola, R., Chen, B., and Zwicker, M., editors, *Eurographics Symposium on Point-Based Graphics*. The Eurographics Association.

Smart, W. D. and Pack Kaelbling, L. (2002). Effective reinforcement learning for mobile robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 4, pages 3404–3410 vol.4.

Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition.

Wang, L., Wang, G., and Alexander, C. A. (2015). Big data and visualization: Methods, challenges and technology progress. *Digital Technologies*, 1(1):33–38.

Watkins, C. J. C. H. and Dayan, P. (1992). Technical note: q-learning. *Mach. Learn.*, 8(3-4):279–292.